Universidad Tecnológica de Pereira

Facultad de Ingenierías

Maestría en Ingeniería de Sistemas y Computación

Clasificación de series de tiempo empleando funciones de distribución y Máquinas de Vectores de Soporte

Christian Montoya Holguin

(TRABAJO DE GRADO DE MAESTRÍA)

Director: Mauricio Alexander Álvarez López

Pereira–2017

Índice general

Capítulo 1

Preliminares

1.1. Planteamiento del problema

La clasificación de series de tiempo a partir de métodos de representación basados en características presenta su dificultad en el momento de seleccionar las técnicas de representación adecuadas para distintos dominios de aplicación, al no ser obvio si las características seleccionadas por un investigador son las mejores características con las que distinguir o representar las clases de datos conocidas debido a que los mecanismos subyacentes a los datos no son bien comprendidos; por lo tanto no es suficiente para muchos problemas prácticos transformar un problema temporal a uno estático donde se transforma una serie de tiempo de cualquier longitud en vectores cortos que encapsulan sus propiedades.

En el presente trabajo se plantea realizar la clasificación de series de tiempo, sin la necesidad de extraer características de las variables de entrada, tratando de encontrar la dependencia entre ellas a través de sus distribuciones de probabilidad como elementos del espacio de Hilbert (RKHS).

1.2. Antecedentes y Justificación

Las series de tiempo son una clase importante de datos presentes en un gran número de dominios de aplicación. Una serie de tiempo $T = t_1, ..., t_m$ es una colección de observaciones hechas cronológicamente [1–4], las cuales además de tener un carácter numérico y continuo, se consideran siempre como un conjunto en lugar de un campo numérico individual. Estas observaciones que se desarrollan con el tiempo normalmente representan información sujeta a análisis, clasificación, indexación, predicción e interpretación [5]. Las series de tiempo se crean rutinariamente en los dominios, científicos, financieros, industriales, de entretenimiento, médicos y biológicos [6]. Aplicaciones como trazas de procesos dinámicos, fluctuaciones diarias en el

mercado de valores, comercio electrónico, mediciones en refinerías de petróleos o plantas químicas, sistemas de reconocimiento de voz, análisis de la marcha de una persona, electrocardiogramas, electroencefalogramas, actualizaciones de posición de objetos móviles en servicios basados en localización, recogen grandes cantidades de datos en forma de series de tiempo. Como consecuencia de esta generación a escala y tasa sin precedentes de este tipo de datos, combinado con dispositivos de computo con una potencia de cálculo superior (giga-FLOPS hasta peta-FLOPS), en las últimas décadas se ha producido una gran cantidad de trabajos de investigación, introduciendo nuevas metodologías para la indexación, agrupación, clasificación y aproximación de series de tiempo.

La exploración de datos en forma de series de tiempo, presenta enormes desafíos debido a su naturaleza de grandes tamaños de datos (usualmente entre cientos y decenas de miles [7]), alta dimensionalidad, constante actualización y ruido en la medición [3] [4]. Esto hace que el tratamiento de series de tiempo sea una tarea difícil para los algoritmos de Machine Learning existentes. En el tratamiento de estas series, entre las necesidades de análisis como se mencionaba anteriormente se encuentran la indexación, el agrupamiento, la clasificación y la aproximación, entre las cuales, la clasificación es ciertamente la más prominente [8] [9] [10]. En el aprendizaje supervisado, a un nuevo patrón se le asigna una etiqueta de clase basada en un conjunto de entrenamiento cuyas etiquetas de clase ya son conocidas.

La clasificación de los datos de las series de tiempo debe tratarse de manera diferente, ya que los datos están en forma de una secuencia de valores que se ordenan en un intervalo de tiempo, es decir los datos se consideran como un conjunto en lugar de un campo numérico individual.

Para enfrentar los retos mencionados, varios investigadores han producido diferentes trabajos, entre los cuales:

En [11] plantean dos aspectos claves para el logro de objetivos en la gestión de datos de series de tiempo, primero los métodos de representación y segundo las medidas de similitud. Los métodos de representación buscan reducir la dimensionalidad de las series preservando las características fundamentales del conjunto de datos, pues al trabajar directamente con estos datos en su formato sin procesar, resulta costoso en términos de coste computacional y de almacenamiento [7]. En cuanto a las medidas de similitud, la distancia entre las series de tiempo necesita ser cuidadosamente definida para capturar adecuadamente la semántica y que refleje la similitud subyacente de tales datos.

Se han propuesto varias técnicas para representar las series de tiempo basadas en características: Transformada Discreta de Fourier (DFT) [12], Descomposición en Valores Singulares (SVD) [12], Transformada Discreta del Coseno (DCT) [13], Transformada Discreta Wavelet (DWT) [14], Aproximación Agregada por Piezas (PAA) [15], Aproximación Constante por Piezas Adaptativa (APCA) [16], Polinomios de

Chebyshev (CHEB) [17], aproXimación Agregada Simbólica (SAX) [18], Aproximación Lineal por Piezas Indexable (IPLA) [19], Características de Texturas de Patrones Recurrentes (TFRP) [10], Representación de Bolsa de Patrones (BoP) [20], en [21] Nanopoulos utilizó la media, la desviación estándar, la asimetría y la curtosis para representar y clasificar las series de tiempo. Wang en [22] por ejemplo, introduce un conjunto de 13 características que contiene medidas de tendencia, estacionalidad, periodicidad, correlación serial, asimetría, curtosis, caos, no linealidad y auto similitud para representar las series. En [23] utilizaron las mediciones de la media, la propagación y la tendencia en intervalos locales de la serie de tiempo para clasificar los diferentes tipos de series de tiempo y por último en [24] presentan un enfoque que se ha extendido desde entonces a series de tiempo multivariadas.

En conjunto con las anteriores técnicas, existen más de una docena de medidas de distancia usadas para evaluar la similaridad entre series de tiempo: Distancia Euclidiana (ED) [12], Distorción de Tiempo Dinámico (DTW) [25, 26], Distancia basada en la Subsecuencia Común Más Larga (LCSS) [27], distancia Editada con Penalidad Real (ERP) [28], Distancia Editada en secuencia Real (EDR) [29], DISSIM [30], Modelo de Alineación con Secuencia Ponderada (Swale) [31], Distancia con Ensamble Espacial (SpADe) [32].

La representación de series de tiempo basada en características presenta problemas a la hora de escoger que tipo de representación es la adecuada para el problema de interés. Por ejemplo en [33] plantean que lo común en DWT y DFT es escoger solamente los primeros coeficientes como características, pero al usar solamente estos coeficientes no se representa de la mejor manera la serie de tiempo. Usando en su lugar los coeficientes más grandes, se conserva la cantidad óptima de energía presente en la señal original, sin embargo, cuando se trata de varias series de tiempo, esto introduce mayores demandas de almacenamiento y/o gastos en los cálculos de distancia. Estas desventajas hicieron Wu descartara esta técnica, al menos para la tarea de indexación [34].

En [35] se propone un método que utiliza una subsecuencia especificada por el usuario de las series de tiempo como conocimiento de previo. El método encuentra patrones similares a partir de datos de series de tiempo mediante el uso de un lenguaje de consulta visual. El lenguaje de consulta se ocupa de dos patrones especificados por el usuario y su combinación. Utilizando este método, los analistas pueden descubrir patrones de características que estén de acuerdo con sus intereses. Sin embargo el método requiere conocimientos básicos dependiendo de las tareas de análisis. Si el conocimiento de los analistas sobre el problema es insuficiente, el método no puede descubrir los patrones de interés.

En [36] plantean que la transformación de un problema temporal a uno estático donde se transforma una serie de tiempo de cualquier longitud en vectores cortos que encapsulan sus propiedades a menudo no es suficiente para muchos problemas prácticos, por ejemplo, el problema de reconocimiento de gestos de la mano, en el que las características

temporales son la posición de las manos, las curvas de los dedos etc. Mirar cualquiera de ellos en un momento aislado no es probable que conduzca a una clasificación exitosa. El reconocimiento de los gestos sólo es posible analizando los cambios de forma de la mano en el dominio del tiempo. En el diagnóstico de fallas de la ingeniería automotriz, los síntomas defectuosos se manifiestan a menudo en varios segmentos de la señal, pero no en puntos aislados.

Lo expuesto permite observar problemas que se presentan en la representación de series de tiempo basada en características. Es difícil escoger que tipo de representación es la adecuada para el problema de interés y usualmente es un investigador o un experto el encargado de seleccionar manualmente las características de un conjunto de datos dado, sin embargo, no es obvio que las características seleccionadas por un investigador determinado sean las mejores características con las que distinguir las clases de datos conocidas, quizás existan alternativas más sencillas con un mejor rendimiento de clasificación. Además, para muchas aplicaciones, los mecanismos subyacentes a los datos no son bien comprendidos, haciendo difícil desarrollar un buen conjunto de características para la clasificación [37].

Por lo anterior se propone una metodología que tome las series de tiempo en su estructura original, que los datos, las variables, entren en bruto al algoritmo de clasificación sin existir la necesidad de extraer, seleccionar o escoger características de las series de tiempo. Al realizar la clasificación de las series de tiempo en su estructura original explotando la dependencia entre las variables se evitan los problemas mencionados en relación a la dificultad de escoger una técnica de representación, la dificultad de escoger que tipo de características extraer de la serie, la necesidad de un experto que realice esta selección y la transformación de un problema temporal a uno estático.

Los métodos kernel son una familia de algoritmos usados ampliamente en aprendizaje de máquina [38]. Su popularidad se puede atribuir a la sólida base matemática dentro de los espacios de Hilbert generados por kernels y porque han demostrado tener buen desempeño en la solución de problemas no lineales. Estos métodos kernels consisten en transformar mediante un operador no lineal, el modelo definido en un espacio de entrada a un espacio de una mayor dimensión la cual puede ser incluso infinita, siendo muy interesante debido a que en una dimensión infinita, siempre se puede clasificar linealmente cualquier conjunto de datos. Este espacio de mayor dimensión es un espacio de Hilbert generado por una función kernel. La importancia de estos modelos radica en no necesitar definir explícitamente la transformación no lineal y la función kernel puede ser expresada como un producto escalar de la transformación no lineal en el espacio de Hilbert [39]. Debido a estas propiedades, los métodos kernel representan una alternativa a los métodos tradicionales no lineales como las redes neuronales artificiales.

Dentro de los métodos kernel, se ha desarrollado principalmente desde el año 2007 el método embebimiento de distribuciones de probabilidad en espacios de Hilbert con

kernel reproductivo (RKHS) [40], el cual ha sido usado para mapear distribuciones en un espacio de Hilbert. La idea de este método es mapear distribuciones de probabilidad en un espacio de alta dimensión, representando las distribuciones de probabilidad como puntos en un RKHS a través de un operador inyectivo. Este método se caracteriza por ser una generalización de los métodos tradicionales basados en kernel [41] y ser muy flexible cuando se aplica a modelos estadísticos de alta dimensión [42] convirtiéndose en una poderosa herramienta para la elaboración de modelos probabilísticos, inferencia estadística y aprendizaje de máquina.

El método embebimiento de distribuciones de probabilidad en un RKHS ha sido empleado en distintas aplicaciones, por ejemplo: en la estimación de la distancia entre distribuciones de probabilidad llamada Máxima Discrepancia en Media MMD [42], en la estimación y predicción de un sistema dinámico [41] en kernel Análisis de Componentes Principales ACP [43], en reducción de dimensión [44], en pruebas de independencia [45]. El método embebimiento de distribuciones de probabilidad en un RKHS ha permitido resolver problemas en procesos aleatorios en espacios de alta dimensión de forma eficiente.

Para las aplicaciones anteriores es necesario tener un estimador consistente de la medida de distancia entre distribuciones de probabilidad para obtener un buen rendimiento, pero las medidas de distancia entre distribuciones de probabilidad y entre procesos aleatorios que se usan en estadística no son verdaderas medidas de distancia: algunas no cumplen la desigualdad triangular o no cumplen la propiedad de simetría. En [46] se presenta una verdadera medida de distancia entre distribuciones de probabilidad llamada MMD. La idea de esta medida de distancia es evaluar la función en muestras empíricas de las distribuciones de probabilidad y observar si las distribuciones de las que han sido extraídas estas muestras son las mismas distribuciones o diferentes.

El método de embebimiento al ser una generalización de los métodos kernel los cuales permiten trabajar con una amplia gama de tipos de datos, al realizar la transformación del espacio de entrada a un espacio de alta dimensión, es una alternativa para la clasificación de series de tiempo al trabajar con las distribuciones de probabilidad de la serie sin la necesidad de estimar sus densidades como un paso intermedio, permitiendo modelar los datos sin necesidad de hacer suposiciones sobre el tipo de distribuciones de probabilidad. El método de embebimiento permite la creación de medidas de similaridad que permiten comparar las distribuciones en el espacio de Hilbert y realizar su clasificación.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar dos algoritmos de clasificación de funciones de distribución empleando máquinas de vectores de soporte.

1.3.2. Objetivos específicos

- Desarrollar una metodología basada en máquinas de vectores de soporte para la clasificación de funciones de distribución.
- Desarrollar una metodología para la estimación de los parámetros de las máquinas de vectores de soporte para la clasificación de funciones de distribución.
- Validar la metodología desarrollada en problemas de clasificación de series de tiempo.

Capítulo 2

Marco teórico

2.1. Aprendizaje supervisado y Máquinas de Vectores de Soporte

El aprendizaje supervisado tiene como objetivo principal utilizar datos de entrenamiento con clases etiquetadas para modelar y clasificar el resultado de datos aún no observados. El proceso se estructura en dos pasos, primero se aprende el modelo usando datos de entrenamiento y luego se prueba el modelo aprendido sobre datos no observados para evaluar la precisión del modelo.

Las Máquinas de Vectores de Soporte (SVM) son una clase importante de algoritmo de aprendizaje supervisado introducido por Vapnik [47] en la década del 90 basado en el concepto de margen. El margen se define como la distancia más pequeña entre el límite de decisión y cualquiera de las muestras, como se observa en la figura 2.1, la cual da una representación del límite de decisión de margen máximo, donde se maximiza la distancia entre los vectores de soporte al límite de decisión conocido.

Dado un conjunto de datos de entrenamiento etiquetados de la forma $\{(y_i, x_i)\}_{i=1}^{N}$, con $y_i \in \{-1, +1\}$ la forma estándar de SVM encuentra el límite de decisión o hiperplano que mejor separa los datos al minimizar un problema de optimización con restricciones:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \tag{2.1}$$

$$sujeto\ a : y_i((wx_i) + w_0) + \xi_i \geq 1$$
$$\xi_i \geq 0$$

donde ξ_i son las variables de holgura y $C > 0$ es el parámetro para compensar la penalización por variables de holgura y el margen.

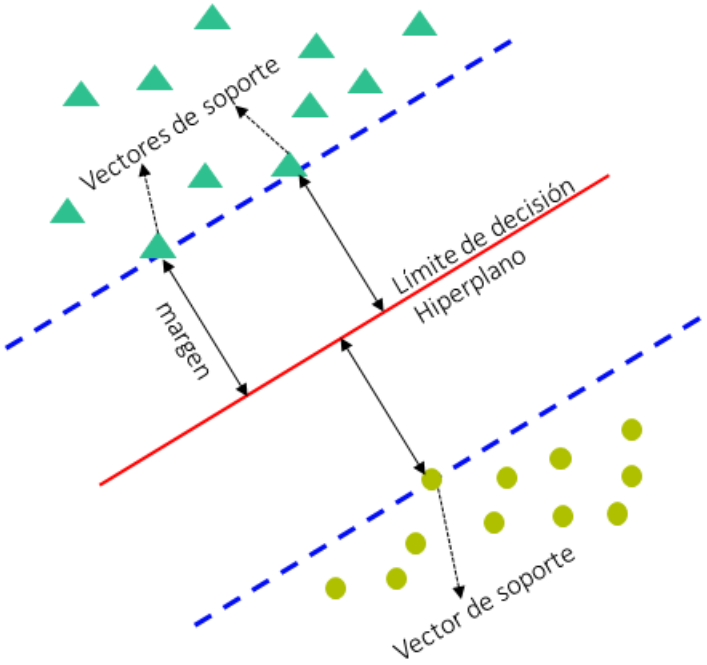


Figura 2.1: Límite de decisión óptimo a partir del máximo margen a los vectores de soporte sobre la línea punteada.

El fundamento de SVM radica en que los límites de decisión no lineales se pueden aprender utilizando el truco del kernel. Un kernel es una función $\mathcal{K} : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, tal que para todo x_i $i \in \{1, ..., N\}$ produce una matriz K simétrica semidefinida positiva (PSD), donde $K_{ij} = k(x_i, x_j)$. La función kernel mapea implícitamente sus entradas a un espacio de alta dimensión conocido comúnmente como espacio de características, $x \mapsto \phi(x)$.

El problema dual consiste en maximizar:

$$\tilde{\alpha} = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{2.2}$$

$$sujeto\ a : 0 \leq \alpha_i \leq C,\ i = 1, ..., N$$

$$\sum_{i=1}^{N} \alpha_i\ y_i = 0$$

La función de decisión es dada por:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i k(x, x_i) + w_0 \tag{2.3}$$

Donde el valor de w_0 se define como:

$$w_0 = \frac{1}{N_S} \sum_{i \in S} (y_i - \sum_{j \in S} \alpha_j y_j k(x_i, x_j)) \tag{2.4}$$

donde N_S es el número total de vectores de soporte [48].

2.2. Embebimiento de distribuciones

Los métodos de embebimiento de distribuciones con kernel mapean las distribuciones a espacios de características de mayor dimensión la cual puede ser incluso infinita, sin la necesidad de construir explícitamente los espacios de características reduciendo los cálculos a las operaciones de la matriz Gram. Se denota por X, a la variable aleatoria con dominio $\mathcal{X}$ y distribución *P(X)*, y las instancias de X por x. Similarmente se denota la variable aleatoria Y con distribución *P(Y)*.

Un espacio de Hilbert con kernel reproductivo (RKHS) $\mathcal{F}$ sobre $\mathcal{X}$ con kernel k es un espacio de Hilbert de funciones $f : \mathcal{X} \mapsto \mathbb{R}$ con producto interno $\langle \cdot, \cdot \rangle_{\mathcal{F}}$. Sus elementos $k(x, \cdot)$ satisfacen la propiedad de reproductiva [42]:

$$\langle f(\cdot), k(x, \cdot) \rangle_{\mathcal{F}} = f(x), \tag{2.5}$$

$$\langle k(x, \cdot), k(x', \cdot) \rangle_{\mathcal{F}} = k(x, x'), \tag{2.6}$$

Significa que se puede observar el mapeo lineal de una función f sobre $\mathcal{X}$ para sus valores en x como un producto punto y el operador de evaluación lineal es dado por $k(x, \cdot)$, la función kernel. Esta función kernel puede ser vista como un mapeo al espacio de características $\phi(x)$ donde $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{F}}$.

Entre funciones kernel comunes sobre $\mathbb{R}^n$ se encuentran el kernel Gaussiano RBF $k(x, x') = exp(-\gamma \|x - x'\|^2)$, kernel de Laplace $k(x, x') = exp(-\lambda \|x - x'\|)$ y el kernel polinomial $k(x, x') = \langle x, x' \rangle^d$.

Los siguientes mapeos son claves en el enfoque de embebimiento [42]:

$$\mu_X := E_X[\phi(X)] = E_X[k(x, \cdot)] \tag{2.7}$$

$$\hat{\mu}_X := \frac{1}{m} \sum_{i=1}^{m} k(x_i, \cdot) \tag{2.8}$$

Donde $X = \{x_1, ..., x_m\}$ se asume que se generan independiente e identicamente distribuidos $i.i.d$ de $P(X)$. Si la condición $E_X[k(x, x)] < \infty$ se satisface, entonces μ_X es un elemento del espacio de Hilbert [42]. Dada la propiedad reproductiva de $\mathcal{F}$, ambos mapeos satisfacen $\langle \mu_X, f \rangle_{\mathcal{F}} = E_X[f(X)]$ y $\langle \hat{\mu}_X, f \rangle_{\mathcal{F}} = \frac{1}{m} \sum_{i=1}^{m} f(x_i)$, lo que indica que es posible calcular la esperanza de la media y su estimador con respecto a $P(X)$ y X respectivamente, tomando el producto interno de los embebimientos μ_X y $\hat{\mu}_X$.

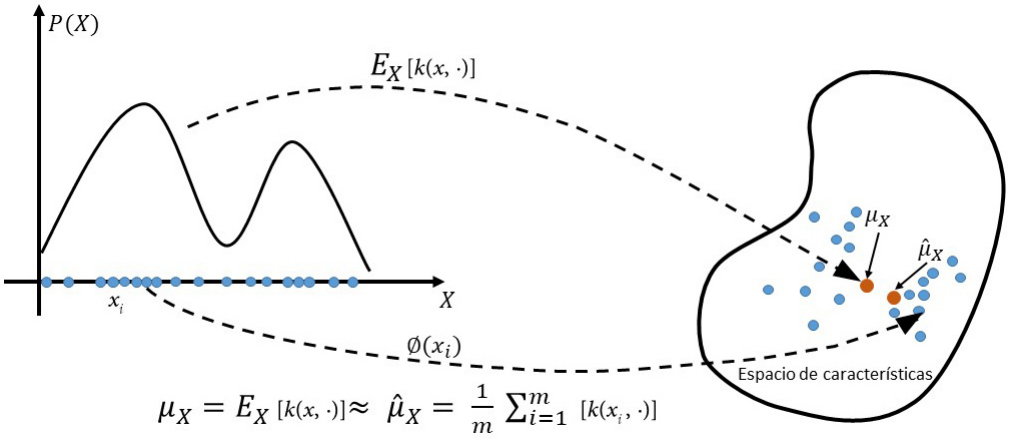


Figura 2.2: Operador de embebimiento de una distribución y su estimador con muestras finitas. Imagen tomada de [49] y editada por el autor.

La figura 2.2 ilustra la esperanza del embebimiento μ_X y su estimador $\hat{\mu}_X$. Debido a que raras veces se tiene acceso a la verdadera distribución, un conjunto de muestras finitas de tamaño m de la distribución $P(X)$ es suficiente para calcular el estimador el cual converge con un error de $O_p(m^{-\frac{1}{2}})$ [49].

2.2.1. MMD

Dada una distribución de probabilidad $P(X)$ definida sobre un conjunto no vacío X, la media embebida de $P(X)$, $\mu_{P(X)} = E_{X \sim P(X)} k(\cdot, X)$, es un elemento de el RKHS $\mathcal{H}_k$ definido por el kernel $k : X \times X \to \mathbb{R}$. Para dos distribuciones de probabilidad $P(X)$

y $P(Y)$, *maximum mean discrepancy* (MMD) [50] entre $P(X)$ y $P(Y)$ esta definida como:

$$\begin{aligned} MMD^2(P(X), P(Y)) &= \|\mu_{P(X)} - \mu_{P(Y)}\|^2_{\mathcal{H}_k} \\ &= E_X E_{X'} k(X, X') + E_Y E_{Y'} k(Y, Y') - 2E_X E_Y k(X, Y) \end{aligned} \tag{2.9}$$

Con X,$X' \overset{i.i.d}{\sim} P(X)$ y Y,$Y' \overset{i.i.d}{\sim} P(Y)$. Dadas las muestras $\{x_i\}_{i=1}^{n_X} \overset{i.i.d}{\sim} P(X)$ y $\{y_j\}_{j=1}^{n_Y} \overset{i.i.d}{\sim} P(Y)$, un estimador de MMD insesgado puede calcularse como:

$$\begin{aligned} \widehat{MMD}^2(P(X), P(Y)) = &\frac{1}{n_X(n_X-1)} \sum_{i=1}^{n_X} \sum_{i' \neq i} k(x_i, x_{i'}) + \\ &\frac{1}{n_Y(n_Y-1)} \sum_{j=1}^{n_Y} \sum_{j' \neq j} k(y_j, y_{j'}) - \frac{2}{n_X n_Y} \sum_{i=1}^{n_X} \sum_{j=1}^{n_Y} k(x_i, y_j) \end{aligned} \tag{2.10}$$

2.3. Embebimiento de distribuciones condicionales

A pesar que el embebimiento de distribuciones proporciona una herramienta potente para hacer frente a un número de desafíos de problemas no paramétricos de alta dimensión, aún quedan muchos problemas por abordar al utilizar estos embebimientos para realizar inferencias [49]. En muchos problemas se relacionan diferentes variables, y modelar solo el embebimiento de las distribuciones marginales de cada variable no permite explotar el conocimiento que tienen estas relaciones entre ellas, relaciones de independencia condicional. En el contexto de las series de tiempo, resulta interesante observar la relación o explotar la dependencia que puede existir entre el valor de la serie tomado en un tiempo $T = t_1$ y otro valor tomado en un tiempo $T = t_2$.

El embebimiento de una distribución condicional $P(Y|X)$ es definido como:

$$\mu_{Y|x} := E_{Y|x}[\phi(Y)] \tag{2.11}$$

Dado este embebimiento, la esperanza condicional de una función $g \in \mathcal{F}$ puede ser calculada como $E_{Y|x}[g(Y)] = \langle g, \mu_{Y|x} \rangle_{\mathcal{F}}$ [49]. Esto puede ser comparado con la propiedad de la media embebida en la sección 2.2 donde la esperanza no condicional de una función puede ser escrita como un producto punto con el embebimiento. A diferencia de los embebimientos discutidos en la sección 2.2, un embebimiento de una distribución

condicional no es un único elemento en el RKHS, resulta ser una familia de puntos en el RKHS, cada uno indexado por un valor fijo x de la variable de condicionamiento X. Sólo fijando X a un valor particular x, es posible obtener un único elemento en RKHS, $\mu_{Y|x} \in \mathcal{F}$. Para lograr lo anterior, es necesario definir un operador denotado como $\mathcal{C}_{Y|X}$, que toma como entrada un x y salida un embebimiento [49].

Se tiene entonces

$$\mu_{Y|x} = \mathcal{C}_{Y|X}\phi(x) \tag{2.12}$$

En [49] asumen $E_{Y|\cdot}[g(Y)] \in \mathcal{F}$

$$\begin{aligned} &\mathcal{C}_{Y|X} := \mathcal{C}_{YX}\mathcal{C}_{XX}^{-1}, \text{ y por lo tanto} \\ &\mu_{Y|x} = \mathcal{C}_{YX}\mathcal{C}_{XX}^{-1}\phi(x) \end{aligned} \tag{2.13}$$

La figura 2.3 muestra un embebimiento de distribuciones condicionales y el operador condicional de embebimiento. En la práctica la inversión del operador $\mathcal{C}_{XX}^{-1}$ puede ser reemplazado por el inverso regularizado $(\mathcal{C}_{XX} + \lambda I)^{-1}$.

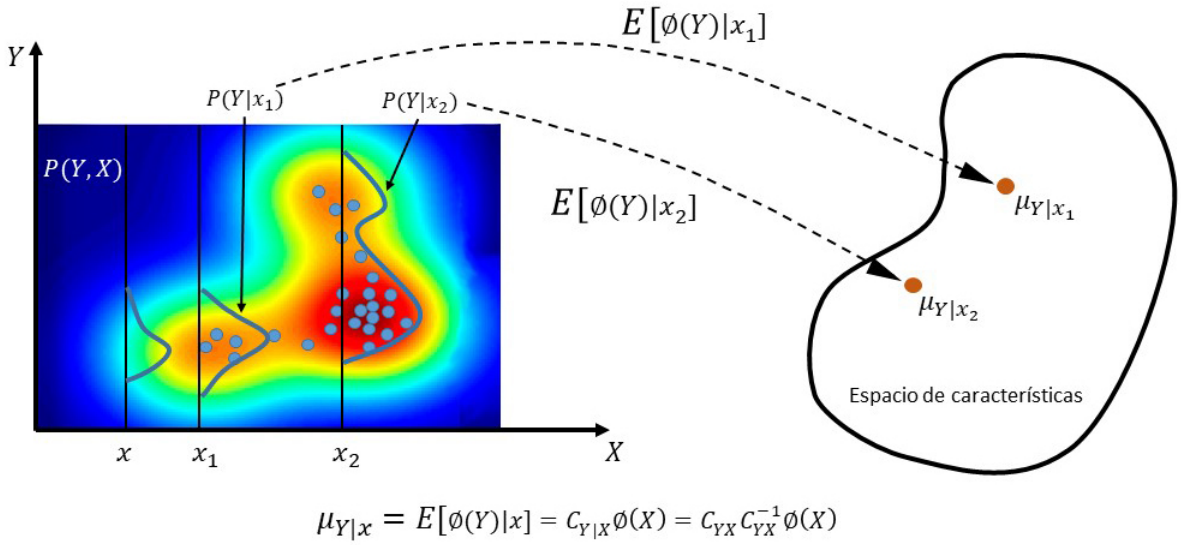


Figura 2.3: Operador de embebimiento de una distribución condicional. Imagen tomada de [49] y editada por el autor

Dado un conjunto de datos $\{(x_1, y_1), ..., (x_m, y_m)\}$ de tamaño m generados $i.i.d$ de $P(X, Y)$ se estima el operador de embebimiento condicional como

$$\hat{\mathcal{C}}_{Y|X} = \Phi(K + \lambda I)^{-1}\Upsilon^T \tag{2.14}$$

Donde $\Phi := (\phi(y_1), ..., \phi(y_m))$ y $\Upsilon := (\phi(x_1), ..., \phi(x_m))$ son matrices de características formadas implicitamente, y $K = \Upsilon^T \Upsilon$ es la matriz Gram para muestras de la variable X. Para evitar el sobreentrenamiento se adiciona el parámetro de regularización λ.

Capítulo 3

Materiales y métodos

Una serie de tiempo $T = t_1, ..., t_{Nl}$ es una secuencia de números reales obtenidos a través de mediciones repetidas a lo largo del tiempo. Como notación en este documento se representa una la serie de tiempo como un conjunto ordenado de Nl variables

$$\{x_n\}_{n=1}^{Nl} \tag{3.1}$$

En la clasificación de series de tiempo empleando funciones de distribución, el objetivo es establecer una relación funcional entre las distribuciones de probabilidad de las series de tiempo de entrada y sus etiquetas de clase como salidas. A partir de un conjunto de entrenamiento $\{(y_l, P_l)\}_{l=1}^{L}$, donde L corresponde al número de realizaciones o cantidad series de tiempo, P_l la distribución de probabilidad de la serie de tiempo y y es la salida que representa la etiqueta de clase correspondiente, $y_l \in \{1, ..., c\}$ siendo c el número de clases. La tarea de la clasificación es asignar a una nueva serie de tiempo en consulta P_{L+1} una etiqueta de clase conocida $y \in \{1, ..., c\}$. Como algoritmo de clasificación se trabaja las Máquinas de Vectores de Soporte (SVM), las cuales son comúnmente empleadas para clasificar representaciones de datos de altas dimensiones construyendo hiperplanos en un espacio multidimensional que separan las diferentes etiquetas de clase [3].

Al no observar directamente las distribuciones de probabilidad, se trabaja con sus muestras i.i.d $P_l \overset{i.i.d}{\sim} x_l^{(1)}, ..., x_l^{(N_l)}$ siendo $\mathcal{X}$ el espacio de muestras subyacente. El conjunto entrenamiento de las series de tiempo y sus etiquetas de clase se representa entonces $\{(y_l, \{x_l^{(n)}\}_{n=1}^{N_l})\}_{l=1}^{L}$, siendo N_l la longitud de las series. Se busca entonces que para una nueva serie de tiempo con muestras $\{x_{L+1}^{(n)}\}_{n=1}^{N_l} \overset{i.i.d}{\sim} P_{L+1}$ se asigne una etiqueta de clase y.

3.1. Clasificador con MMD

Se utiliza MMD como medida de similaridad, la cual permite comparar en un RKHS si dos conjuntos de datos pertenecen a la misma distribución de probabilidad. En adelante este método de clasificación se nombrará SVM-KMMD.

Primero se realiza el embebimiento de las distribuciones a partir de las muestras i.i.d de la serie de tiempo $x_1, ..., x_{Nl}$, vía la media embebida μ dentro de RKHS $\mathcal{H}_k$ definido por el kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, que para este caso es un kernel RBF $k(x, x') = exp(-\gamma \|x - x'\|^2)$ con $\gamma = \frac{1}{2\sigma^2}$.

Dada las muestras de una serie de tiempo A, $\{x_i^a\}_{i=1}^{n_A}$, dibujadas de su distribución de probabilidad P_A y las muestras de una serie de tiempo B, $\{x_j^b\}_{j=1}^{n_B}$, dibujadas de su distribución de probabilidad P_B. Se calcula el estimador insesgado de MMD

$$\widehat{MMD}^2(P_A, P_B) = \frac{1}{n_A(n_A-1)} \sum_{i=1}^{n_A} \sum_{i' \neq i} k(x_i^a, x_{i'}^a) + \frac{1}{n_B(n_B-1)} \sum_{j=1}^{n_B} \sum_{j' \neq j} k(x_j^b, x_{j'}^b) - \frac{2}{n_A n_B} \sum_{i=1}^{n_A} \sum_{j=1}^{n_B} k(x_i^a, x_j^b) \tag{3.2}$$

Segundo, el resultado $\widehat{MMD}^2$ de la etapa anterior se compone en un segundo kernel de espacio RKHS $\mathcal{H}_K$

$$K(\mu_{\widehat{P_l}}, \mu_{\widehat{P_{l'}}}) = exp(-\gamma_K \widehat{MMD}^2(P_l, P_{l'}))$$
$$\gamma_K = \frac{1}{2\sigma_K^2} \tag{3.3}$$

El cual provee una medida de similaridad sobre el espacio de medias embebidas.

Tercero se realiza la clasificación en el RKHS $\mathcal{H}_K$ implementando máquinas de vectores de soporte SVM con el kernel K de (3.3).

$$f(\mu_n) = \sum_{l=1}^{L} \alpha_l y_l K(\mu_n, \mu_l) + w_0 \tag{3.4}$$

Se presentan 3 parámetros para esta metodología de clasificación, tabla (3.1):

Descripción	Parámetro
Sigma kernel RBF para el cálculo de MMD	σ
Sigma kernel compuesto con MMD	σ_K
Parámetro SVM	C

Tabla 3.1: Parámetros clasificador con MMD, SVM-KMMD.

3.2. Clasificador de distribuciones condicionales

En las distribuciones condicionales se asume que solo ciertos aspectos en los datos tienen una influencia directa en la tarea de clasificación. Por lo tanto la atención se centra en modelar estos aspectos estableciendo una relación funcional entre las distribuciones condicionales de las series de tiempo y sus etiquetas de clase. Este método de clasificación recibe el nombre de SVM-KDC.

Primero Se descomponen los datos de las series de tiempo

$$\begin{aligned} x_l^{(n)} &= (z_l^{(n)}, w_l^{(n)}) \\ w_l^{(n)} &\rightarrow \text{Codifica aspectos importantes en los datos} \\ z_l^{(n)} &\rightarrow \text{Describe el resto de información en los datos} \end{aligned} \tag{3.5}$$

Se asume que y_l depende de P_l solo a través de las condicionales inducidas $\{P_l(\cdot|z_l^{(n)})\}_{n=1}^{N_l}$). Para asegurar que todos los objetos matemáticos existan y estén bien definidos, se siguen las suposiciones de [51].

Se tiene entonces una serie de tiempo l de longitud N_l descompuesta en términos de (z, w)

$$\{x_l^{(n)}\}_{n=1}^{N_l} = \{(z_l^{(n)}, w_l^{(n)})\}_{n=1}^{N_l}$$

Por ejemplo, en la ecuación 3.6 se descompone la serie de tiempo de la figura 3.1 en términos de z y w. Para obtener estos valores se tiene una variable de holgura $\tau \in \{1, 2, ..., N_l - 1\}$ que representa la distancia entre muestras que conforman la dupla (z, w).

De la serie de tiempo de la figura 3.1 con un $\tau = 1$ se tiene

$$\begin{aligned} \{x_l^{(n)}\}_{n=1}^{N_l-\tau} &= (\{z_l^{(n)}\}_{n=1}^{N_l-\tau}, \{w_l^{(n)}\}_{n=1}^{N_l-\tau}) \\ &= (z_l^{(1)}, w_l^{(1)}), (z_l^{(2)}, w_l^{(2)}), ..., (z_l^{(N_l-\tau)}, w_l^{(N_l-\tau)}) \end{aligned} \tag{3.6}$$

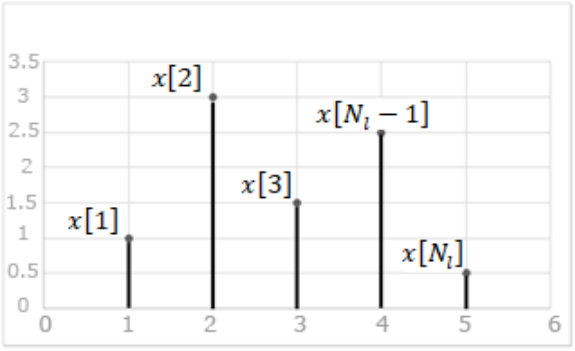


Figura 3.1: Serie de tiempo

Donde $\{z_l^{(n)}\}_{n=1}^{N_l-\tau} = x[n]_{n=1}^{N_l-\tau}$ y $\{w_l^{(n)}\}_{n=1}^{N_l-\tau} = x[n]_{n=1+\tau}^{N_l}$ resultando las duplas

$$\begin{aligned} x_l^{(1)} &= (x[1], x[2]) \\ x_l^{(2)} &= (x[2], x[3]) \\ &\vdots \\ x_l^{(N_l-\tau)} &= (x[N_l-1], x[N_l]) \end{aligned}$$

Segundo para cada serie de tiempo l se codifica el embebimiento de la familia inducida de distribuciones condicionales $\{P_l(\cdot|z_l^{(n)})\}_{n=1}^{N_l}$ con el operador de embebimiento condiconal:

$$\mathcal{C}_{W|Z}^{(l)} \tag{3.7}$$

Se calcula el estimador del operador de embebimiento condicional

$$\widehat{\mathcal{C}}_{W|Z}^{(l)} = \mathbf{k}_W^{(l)}(\mathbf{k}_{ZZ}^{(l)} + \lambda I)^{-1}\mathbf{k}_Z^{(l)} \tag{3.8}$$

Con $\mathbf{k}_W^{(l)} = [k_{\mathcal{W}}(\cdot, w_l^{(1)}), ..., k_{\mathcal{W}}(\cdot, w_l^{(N_l)})]$, $\mathbf{k}_Z^{(l)} = [k_{\mathcal{Z}}(\cdot, z_l^{(1)}), ..., k_{\mathcal{Z}}(\cdot, z_l^{(N_l)})]^T$ y $[\mathbf{k}_{ZZ}^{(l)}] = k_{\mathcal{Z}}(z_l^{(i)}, z_l^{(j)})$.

Tercero se define un nuevo RKHS $\mathcal{H}_K$ con kernel

$$K(C, C') = Tr(CC') \tag{3.9}$$

Sus estimadores

$$K(\widehat{C}^{(l)}_{W|Z}, \widehat{C}^{(l')}_{W|Z}) = Tr[\widehat{C}^{(l)}_{W|Z}\widehat{C}^{(l')}_{W|Z}] \tag{3.10}$$

Aplicando la propiedad de la traza $Tr(AB) = Tr(BA)$ se obtiene

$$Tr[\widehat{C}^{(l)}_{W|Z}\ \widehat{C}^{(l')}_{W|Z}] = [(k(z^l, z^l) + \lambda I)^{-1}\ k(z^l, z^{l'})\ (k(z^{l'}, z^{l'}) + \lambda I)^{-1}\ k(w^l, w^{l'})] \tag{3.11}$$

El kernel de la ecuación (3.11) provee una medida de similaridad sobre el espacio de operadores de embebimiento condicional.

Cuarto se realiza la clasificación en el RKHS $\mathcal{H}_K$ implementando máquinas de vectores de soporte SVM con el kernel K de la ecuación (3.11).

$$f(\mu_{n_{W|Z}}) = \sum_{l=1}^{L} \alpha_l y_l K(\mu_{n_{W|Z}}, \mu_{l_{W|Z}}) + w_0 \tag{3.12}$$

Se presentan 4 parámetros para esta metodología de clasificación, tabla (3.2):

Descripción	Parámetro
Tao	τ
Sigma kernel RBF	σ
Lambda	λ
Parámetro SVM	C

Tabla 3.2: Parámetros clasificador de distribuciones condicionales, SVM-KDC

3.3. Conjunto de datos

La evaluación de los dos clasificadores planteados se realiza usando un amplio conjunto de datos de diferentes dominios disponible en *"The UCR Time Series Classification Archive"* [52]. Este es uno de los repositorios más grandes de series de tiempo en el mundo y algunos autores estiman que representa alrededor del 90 % de todos los conjuntos de datos etiquetados disponibles al público [5]. El conjunto de datos se encuentra dividido en datos de entrenamiento y datos de prueba, facilitando los

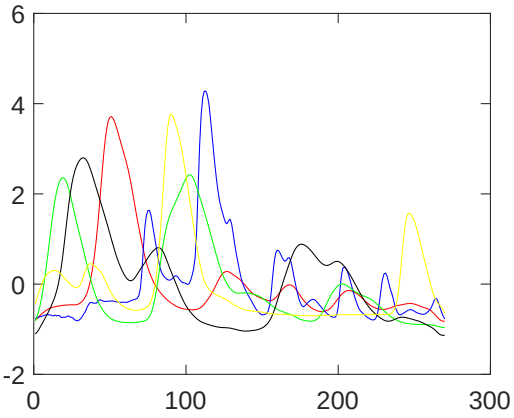
experimentos realizados sobre ellos y la comparación de resultados con otros métodos propuestos en la literatura. El repositorio incluye conjuntos de datos del mundo real, así como datos sintéticos, además de incluir series de tiempo unidimensionales extraídas de datos bidimensionales como imágenes.

Tabla 3.3: Resumen del conjunto de datos de series de tiempo.

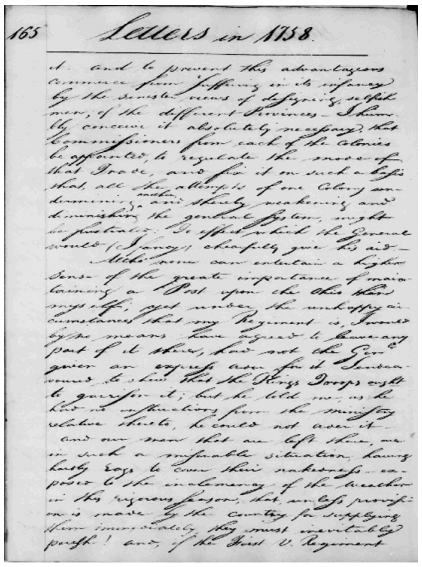
Nombre conjunto	Número de clases	Conjunto de entrenamiento	Conjunto de prueba	Longitud de la serie
50words	50	450	455	270
Adiac	37	390	391	176
Beef	5	30	30	470
CBF	3	30	900	128
Coffee	2	28	28	286
ECG200	2	100	100	96
FISH	7	175	175	463
FaceFour	4	24	88	350
Gun_Point	2	50	150	150
Lighting2	2	60	61	637
Lighting7	7	70	73	319
OSULeaf	6	200	242	427
OliveOil	4	30	30	570
SwedishLeaf	15	500	625	128
Trace	4	100	100	275
synthetic_control	6	300	300	60

La tabla 3.3 contiene el resumen de 16 diversos conjuntos de datos de las series de tiempo utilizadas, las cuales abarcan problemas de dos clases hasta problemas multiclase para una amplia variedad de aplicaciones, por ejemplo, datos biomédicos, espectogramas, medidas electromagnéticas y datos sintéticos de sensores en plantas nucleares. Todos los conjuntos de datos se encuentran normalizados y presentan una escala máxima de 1, además para cada conjunto de datos de series de tiempo, se define un conjunto de series para entrenamiento y un conjunto de series para prueba.

Se realizan algunas descripciones de los conjuntos de datos utilizados para tener una idea general del dominio de aplicación de las series de tiempo estudiadas. El conjunto 50words es un conjunto de datos de contornos de palabras tomadas de la biblioteca George Washington por T. Rath y usadas en el documento [53]. Cada realización es una palabra y la serie se forma tomando el perfil de altura de la palabra, figura 3.2. El conjunto Adiac nace del proyecto de investigación y clasificación automática de



(a) Series de tiempo 50words



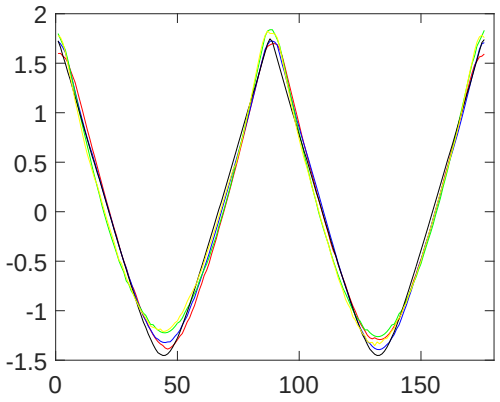
(b) Documento biblioteca George Washington

Figura 3.2: Conjunto 50words. Como ejemplo en 3.2a se representan 5 realizaciones de 5 clases dentro de las 50 clases del conjunto 50words. En 3.2b se representa un documento de la biblioteca George Washington [54].

Diatomeas (algas unicelulares) a partir de imágenes. Los contornos de las Diatomeas se extraen de imágenes con umbral y las series de tiempo se generan como la distancia del contorno a un punto de referencia. En la figura 3.3 se ilustra un ejemplo de este conjunto de datos. El conjunto de datos FaceFour representa los contornos de rostros de cuatro personas, la figura 3.4 ilustra un ejemplo del contorno del rostro de una persona y una serie de tiempo por cada clase dentro del conjunto de datos.

El conjunto OSULeaf se conforma por series de tiempo obtenidas del contorno de imágenes digitalizadas de seis clases de hojas: Arce Circinatum, Arce Glabrum, Arce Macrophyllum, Arce Negundo, Quercus Garryana y Quercus Kelloggii. La figura 3.5 representa este conjunto.

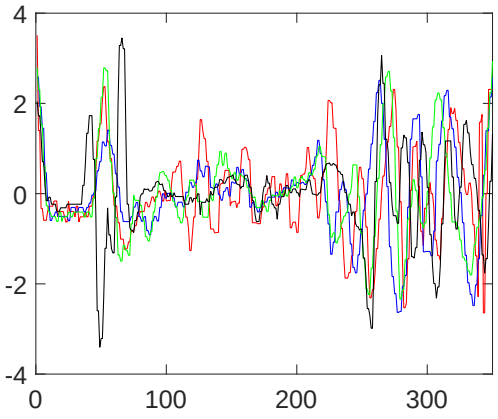
El conjunto Synthetic_control contiene 600 series de tiempo de gráficos de control sintéticamente generados por los procesos en el artículo [58]. Se conforma de seis clases diferentes de gráficos de control: normal, cíclico, tendencia creciente, tendencia decreciente, desplazamiento hacia arriba y desplazamiento hacia abajo, ver figura 3.6.



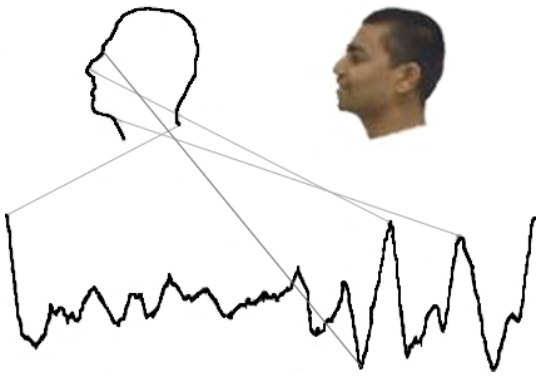
(a) Series de tiempo Adiac

(b) Imágenes de Diatomeas

Figura 3.3: Conjunto Adiac. Como ejemplo en 3.3a se representan 5 realizaciones de 5 clases dentro de las 37 clases del conjunto Adiac. La figura 3.3b presenta diferentes Diatomeas [55].

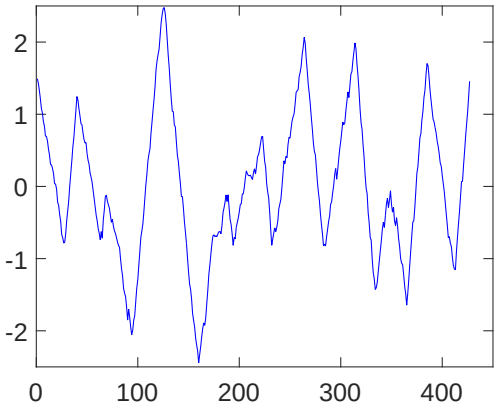


(a) Series de tiempo FaceFour

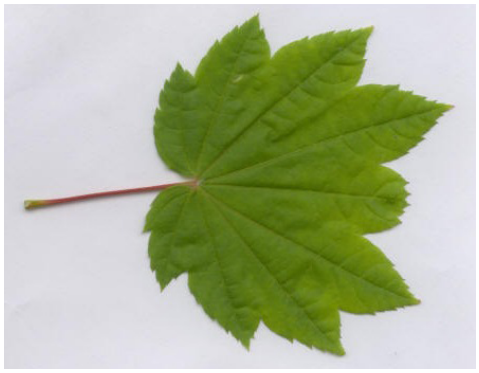


(b) Ejemplo contorno de rostro

Figura 3.4: Conjunto FaceFour. En 3.4a se representa una realización de cada clases. La figura 3.4b presenta el cortorno de un rostro [56].



(a) Serie de tiempo OSULeaf



(b) Ejemplo hoja de Arce Circinatum

Figura 3.5: Conjunto OSULeaf. En 3.5a se representa una realización del conjunto de datos. La figura 3.5b presenta una hoja de Arce Circinatum [57].

3.4. Selección de parámetros

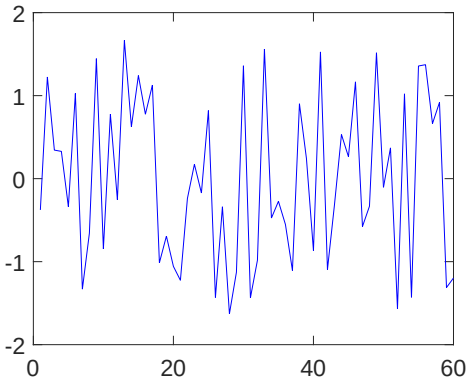
Los parámetros que aparecen en las tablas (3.1) y (3.2) influyen significativamente en el rendimiento de los clasificadores. Por lo tanto el primer paso en los experimentos es encontrar los parámetros que presentan la mejor precisión en la clasificación. Se varía entonces los parámetros usando la técnica de búsqueda en cuadrícula propuesta en [59], en la cual estableciendo los valores mínimos, máximos y incremento de cada parámetro, se evalúa el rendimiento de cada combinación usando el enfoque de validación cruzada en los datos de entrenamiento, y los datos de prueba se restringen a la evaluación final del clasificador.

Se usa la librería LIBSVM [60] para implementar los clasificadores SVM.

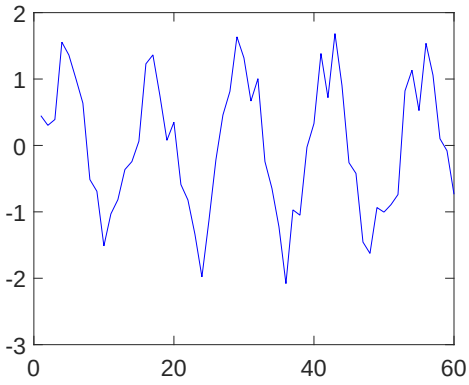
3.5. Librería LIBSVM

LIBSVM es una librería para Máquinas de Vectores de Soporte (SVM) que ha venido en desarrollo desde el año 2000 [60] y actualmente es una de las librerías más usadas para resolver problemas de clasificación y regresión mediante SVM. LIBSVM es código abierto disponible en C++ y Java pero también se ha adaptado para más lenguajes de programación como Python, R, Matlab, Per1, Weka, Ruby, LabView y PHP.

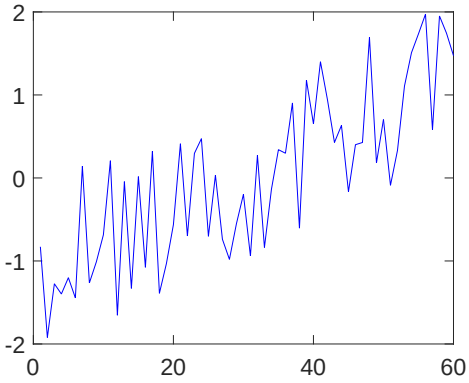
Los algoritmos de entrenamiento y predicción se encuentran en el archivo svm.cpp el cual tiene como subrutinas principales a *svm_train* y *svm_predict*. El procedimiento de entrenamiento es más sofisticado al maximizar el problema de optimización de la



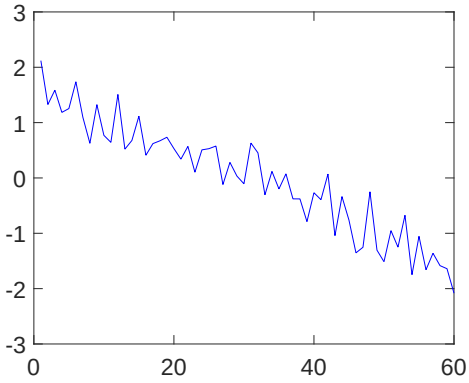
(a) Clase Normal



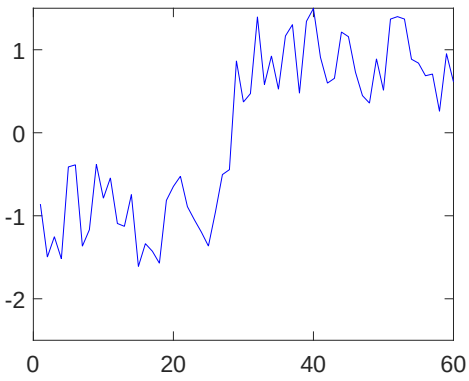
(b) Clase Cíclica



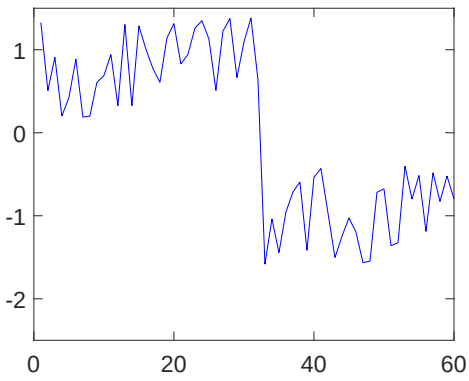
(c) Clase Tendencia Creciente



(d) Clase Tendencia Decreciente



(e) Clase Desplazamiento Arriba



(f) Clase Desplazamiento Abajo

Figura 3.6: Conjunto Synthetic control.

SVM, ecuación 2.2, y desacoplar los problemas multiclase a problemas de dos clases. Para resolver el problema de programación cuadrática, LIBSVM considera un método de descomposición conocido como Optimización Mínima Secuencial (SMO) para tratar la dificultad de matrices densas. SMO consiste en resolver subproblemas del problema inicial para que el coste computacional sea menor.

Como medida de desempeño en el clasificador utiliza

$$\text{Precisión} = \frac{\#\text{ de datos correctamente clasificados}}{\#\text{ total de datos de prueba}} \times 100 \tag{3.13}$$

Para la clasificación multiclase, LIBSVM implementa el enfoque uno contra uno.

3.6. Significancia estadística

Para estudiar si hay diferencias que son estadísticamente significativas entre los rendimientos a nivel general de los clasificadores sobre los conjuntos de datos estudiados, se realiza la prueba Kruskal-Wallis para comparar el rendimiento promedio de los clasificadores sobre los 16 conjuntos de datos. Se realiza esta prueba debido a que el tamaño del conjunto es tal que no se puede determinar con seguridad si los resultados se distribuyen de forma normal y Kruskal-Wallis no requiere asumir normalidad en los datos. Si se rechaza la hipótesis nula para medianas iguales, se realiza una prueba de comparación múltiple utilizando Tukey-Kramer para estudiar más a fondo si existe diferencia entre los rendimientos generales de los clasificadores. Todos los niveles de significancia medidos al 5 %.

3.7. Complejidad computacional

En esta sección se revisa el costo computacional requerido para clasificar las series de tiempos después de haber encontrado los hiperparámetros para cada uno de los métodos propuestos. Dado el pseudocódigo para el clasificador SVM-KMMD, algoritmo 1, presenta un costo computacional $O(N^2L^2 + L^2)$ para un problema de clasificación de L series de tiempo de longitud N. El costo computacional del clasificador SVM-KDC representado en el algoritmo 2 es $O(N^3L^2 + L^2)$.

Algoritmo 1: Clasificador SVM-KMMD

Entrada: Conjunto de series de tiempo etiquetadas
Salida : Tasa de error del clasificador SVM

Búsqueda de parámetros σ, σ_K y C ;
for $\sigma \leftarrow 2^{-9}$ **to** 2^{1} **do**
    Cálculo de $\widehat{MMD}^2$ con kernel RBF y parámetro σ para las L series de tiempo del conjunto de entrenamiento;
    for $C \leftarrow 2^{-5}$ **to** 2^{11} **do**
        for $\sigma_K \leftarrow 2^{-13}$ **to** 2^{3} **do**
            Se calcula el kernel de la ecuación (3.3) del espacio RKHS $\mathcal{H}_K$;
            Se realiza validación cruzada CV para encontrar los parámetros del modelo, clasificador SVM.;
            Selección de parámetros que presentan la menor tasa de error en el proceso de validación cruzada CV;
        end
    end
end
Cálculo de $\widehat{MMD}^2$ con el mejor σ para las L series de tiempo del conjunto de entrenamiento;
Cálculo del kernel de la ecuación (3.3) del espacio RKHS $\mathcal{H}_K$ con el mejor σ_K;
Cálculo del modelo de la SVM con el mejor C;
`// Se valida el modelo con las series de tiempo del conjunto de prueba`;
Cálculo de $\widehat{MMD}^2$ para las series de tiempo del conjunto de prueba con el mejor σ;
Cálculo del kernel de la ecuación (3.3) del espacio RKHS $\mathcal{H}_K$ con el mejor σ_K;
Predicción de la etiqueta de cada serie del conjunto de prueba usando SVM con el mejor valor de C;
return *Tasa de error del clasificador SVM sobre el conjunto de prueba*;

Algoritmo 2: Clasificador SVM-KDC

Entrada: Conjunto de series de tiempo etiquetadas
Salida : Tasa de error del clasificador SVM

Búsqueda de parámetros τ, σ, λ y C ;
$\tau \longleftarrow 1$;
for $\lambda \leftarrow 0{,}5$ **to** 1 **do**
  for $C \leftarrow 2^{-5}$ **to** 2^{13} **do**
    for $\sigma \leftarrow 2^{-15}$ **to** 2^{3} **do**
      Se descomponen los datos de las series de tiempo $x_l^{(n)} = (z_l^{(n)}, w_l^{(n)})$ con $\tau = 1$;
      Cálculo del kernel de la ecuación (3.11);
      Se realiza validación cruzada CV para encontrar los parámetros del modelo, clasificador SVM.;
      Selección de parámetros que presentan la menor tasa de error en el proceso de validación cruzada CV;
    end
  end
end
for $\tau \leftarrow 1$ **to** 4 **do**
  Se descomponen los datos de las series de tiempo $x_l^{(n)} = (z_l^{(n)}, w_l^{(n)})$ con $\tau = 1$;
  Cálculo del kernel de la ecuación (3.11) con los mejores valores de los parámetros λ, σ y C ;
  Se realiza validación cruzada CV para encontrar el mejor τ , clasificador SVM.;
  Selección de τ que presenta la menor tasa de error en el proceso de validación cruzada CV;
end
`// Entrenamiento con los mejores parámetros`;
Se descomponen los datos de las L series de tiempo del conjunto de entrenamiento $x_l^{(n)} = (z_l^{(n)}, w_l^{(n)})$ con el mejor τ;
Cálculo del kernel de la ecuación (3.11) para las L series de tiempo del conjunto de entrenamiento;
Cálculo del modelo de la SVM con el mejor C;
`// Se valida el modelo con las series de tiempo del conjunto de prueba`;
Se descomponen los datos de las series de tiempo del conjunto de prueba $x_l^{(n)} = (z_l^{(n)}, w_l^{(n)})$ con el mejor τ;
Cálculo del kernel de la ecuación (3.11) para las series de tiempo del conjunto de prueba con el mejor σ y λ;
Predicción de la etiqueta de cada serie del conjunto de prueba usando SVM con el mejor valor de C;
return *Tasa de error del clasificador SVM sobre el conjunto de prueba*;

Capítulo 4

Resultados

En este capítulo se presentan los diferentes experimentos realizados y los resultados obtenidos sobre los 16 conjuntos de datos de series de tiempo estudiados. Las simulaciones se implementan en Matlab sobre una computadora Asus K43E con un Intel Core i5 a 2.50GHz.

Los resultados de los clasificadores SVM-KMMD y SVM-KDC se presentan en la tabla 4.1 junto con la tasa de error de dos clasificadores del estado del arte: 1-NN con distancia Euclidiana (ED) [52] y 1-NN con DTW [52]. Estos clasificadores comparten con los métodos propuestos en este trabajo la característica de ser métodos de clasificación basados en instancias. Además se incluyen 3 métodos basados en características o representaciones como FBL [37], BoP [20] y TFRP_20 [10]. En esta investigación no se han implementado estos métodos, los resultados se tomaron de las publicaciones basadas en el conjunto de datos UCR.

Comparando SVM-KDC con los clasificadores basados en instancias SVM-KMMD, 1-NN ED y 1-NN DTW se tiene que el clasificador SVM-KDC es efectivo y presenta bajas tasas de error. Por ejemplo, sobre los 16 conjuntos de datos de series de tiempo, SVM-KDC supera a SVM-KMMD en 12 conjuntos de datos y logra un resultado equivalente sobre 2 conjuntos de datos. El clasificador SVM-KDC supera a 1NN-ED sobre 14 conjuntos de datos y logra un resultado equivalente sobre un conjunto de datos, y al clasificador 1NN-DTW lo supera sobre 7 conjuntos de datos y logra un resultado equivalente sobre un conjunto de datos.

Al comparar SVM-KDC con los métodos basados en características también se observan resultados interesantes. Por ejemplo, sobre los 16 conjuntos de datos de series de tiempo, SVM-KDC supera a FBL en 9 conjuntos y logra un resultado equivalente sobre 2 conjuntos de datos, frente a BoP presenta mejor rendimiento sobre 8 conjuntos y por último, sobre 13 conjuntos de datos SVM-KDC supera a TFRP_20 en 7 conjuntos y logra un resultado equivalente sobre un conjunto de datos.

La superioridad del clasificador SVM-KDC sobre el clasificador SVM-KMMD confirma la importancia de explotar la dependencia entre los datos de la serie de tiempo. El

Tabla 4.1: Estudio comparativo usando los conjuntos de datos de UCR: Tasa de error de los clasificadores

Serie	**SVM KMMD**	**SVM KDC_(t)**	**1NN ED**	**1NN DTW**	**FBL**	**BoP**	**TFRP**
50words	0,701	0,310 (2)	0,369	**0,242**	0,453	0,466	0,481
Adiac	0,220	0,238 (1)	0,389	0,391	0,355	0,432	**0,205**
Beef	0,366	**0,100 (1)**	0,333	0,333	0,433	0,433	0,500
CBF	0,351	0,088 (1)	0,148	**0,004**	0,289	0,013	
Coffee	0,100	**0,000 (1)**	**0,000**	**0,000**	**0,000**	0,036	0,036
ECG200	0,170	0,110 (1)	0,120	0,120	**0,010**	0,150	0,170
FISH	0,320	0,125 (2)	0,217	0,154	0,171	**0,074**	0,154
FaceFour	0,215	0,215 (1)	0,216	0,114	0,261	**0,023**	0,250
Gun_Point	0,110	0,053 (2)	0,087	0,087	0,073	**0,027**	**0,027**
Lighting2	0,213	0,213 (1)	0,246	**0,131**	0,197	0,164	0,180
Lighting7	0,424	0,397 (2)	0,425	**0,288**	0,438	0,466	0,411
OSULeaf	0,525	0,426 (1)	0,479	0,388	0,165	0,256	**0,103**
OliveOil	0,133	**0,100 (1)**	0,133	0,133	**0,100**	0,133	**0,100**
SwedishLeaf	0,230	0,112 (1)	0,211	0,154	0,227	0,198	**0,074**
Trace	0,020	0,140 (1)	0,240	0,010	0,010	**0,000**	
synthetic control	0,510	0,360 (1)	0,120	**0,017**	0,037	0,037	
Media	0,288	0,187	0,233	0,160	0,201	0,182	0,207
Gana	0/16	3/16	1/16	6/16	3/16	4/16	5/16

clasificador SVM-KMMD asume que todos los aspectos en los datos son importantes modelando solo las distribuciones marginales mientras que el clasificador SVM-KDC modela las distribuciones condicionales teniendo en cuenta la dependencia entre los datos, característica importante para trabajar con problemas de datos ordenados, jerárquicos o espacio temporales como los estudiados en este trabajo de grado.

Además de la tabla 4.1 se presentan diagramas de dispersión para comparar pares de métodos y observar su rendimiento. En estos diagramas, cada conjunto de datos es representado por un punto en el plano cartesiano, donde la coordenada x representa la tasa de error obtenida de un método de clasificación y la coordenada y representa la tasa de error obtenida por el método competidor. Por ejemplo en la figura 4.1, los puntos por encima de la línea indican que SVM-KDC presenta menor tasa de error que SVM-KMMD. Entre más lejos se encuentre un punto de la línea, mayor será la tasa de error de un método frente al otro. La mayoría de puntos sobre una región indican cual método presenta menor rendimiento.

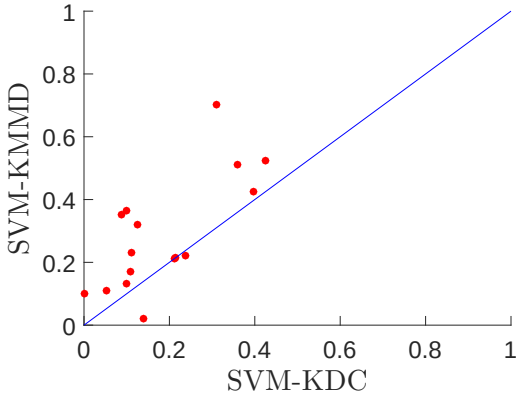
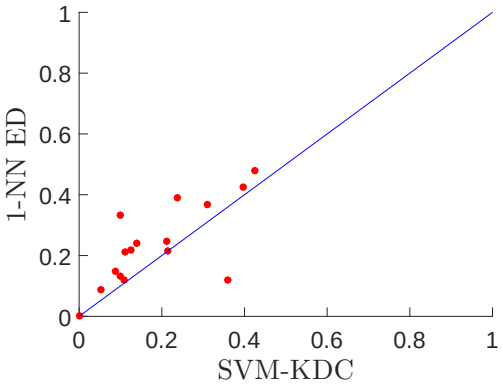


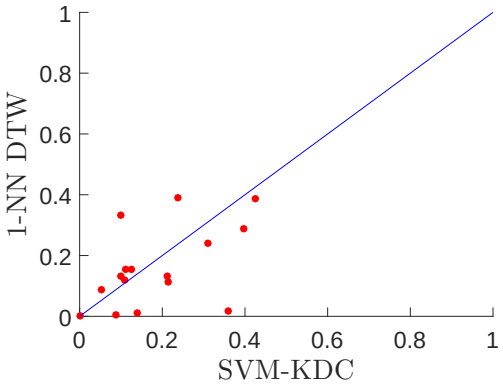
Figura 4.1: Representación gráfica de los resultados logrados por SVM-KDC versus SVM-KMMD.

Los resultados logrados por SVM-KDC frente ED y DTW son altamente competitivos, la importancia de este resultado radica en que estos métodos son considerados estado del arte en la clasificación de series de tiempo y son difíciles de superar.

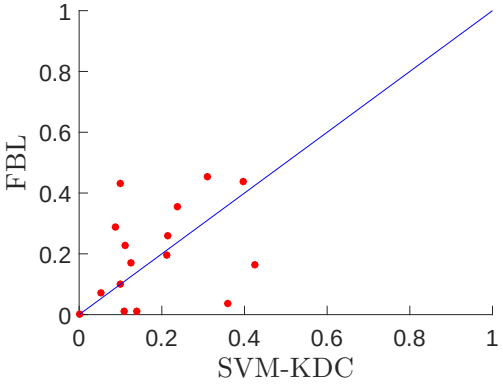
Al observar la figura 4.2c se observa como una gran mayoría de conjuntos de datos presentan menor tasa de error con SVM-KDC respecto a su competidor FBL. Esta comparación es interesante al revisar un método de clasificación basado en instancias frente a un método basado en características. FBL genera un gran conjunto de características para capturar factores discriminatorios en los datos [37], luego las características más informativas se seleccionan usando un clasificador lineal. A diferencia de SVM-KDC, FBL incluye una etapa de preprocesamiento de las series para



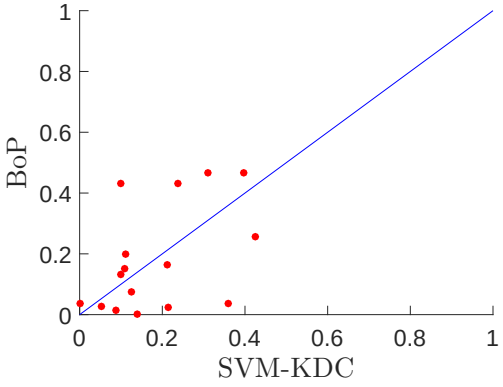
(a) SVM-KDC vs 1-NN ED



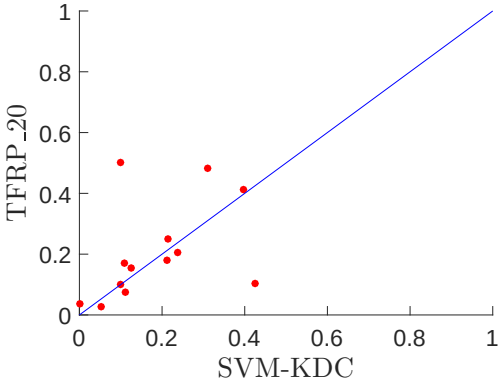
(b) SVM-KDC vs 1-NN DTW



(c) SVM-KDC vs FBL



(d) SVM-KDC vs BoP



(e) SVM-KDC vs TSRP

Figura 4.2: Representación gráfica de los resultados logrados por SVM-KDC versus 1NN-ED, 1NN-DTW, FBL, BoP y TSRP

encontrar sus características discriminatorias y en algunos casos donde el conjunto de datos presenta un gran número de clases y el tamaño de las clases en el conjunto de entrenamiento presenta una gran heterogeneidad, por ejemplo el conjunto 50words que presenta clases desde tan solo 1 hasta 52 ejemplos en el conjunto de entrenamiento, la tarea de seleccionar las características en FBL que mejor capturen las diferencias entre las clases se hace difícil, arrojando altas tasas de error y poniendo en una mejor posición al clasificador SVM-KDC el cual no realiza un preprocesamiento de las series y obtiene una tasa de error menor en este tipo de conjuntos de datos. En conjuntos que presentan pocas series para entrenamiento, y el tamaño de sus clases es bastantes heterogéneo, los métodos basados en características presentan dificultad al seleccionar las características que mejor capturan las diferencias entre las clases. En conjuntos como FaceFour donde ocurre lo antes mencionado, nuestro método SVM-KDC presenta mejores resultados frente a los métodos FBL y TFRP.

La figura 4.2e ilustra que SVM-KDC y TFRP tienen un rendimiento cercano, con cierta ventaja hacia el clasificador SVM-KDC. Se evidencia nuevamente la ventaja que presenta SVM-KDC frente al método basado en características, en conjuntos de datos con heterogeneidad en el tamaño de sus clases y con pocas series para el entrenamiento; esto se observa en los puntos alejados de la línea sobre la región del clasificador TFRP, puntos que representan a los conjuntos de series 50words y Beef.

La figura 4.2e también permite observar un conjunto de datos donde el clasificador SVM-KDC presenta una alta tasa de error. OSULeaf es un conjunto para el que ambos métodos propuestos, SVM-KMMD y SVM-KDC, presentan una alta tasa de error. Además al revisar la tabla 4.1 se encuentra un pobre rendimiento de los métodos propuestos sobre el conjunto Synthetic-Control a diferencia de los otros métodos que presentan bajas tasas de error. Al observar la dinámica de estos conjuntos se encuentra por ejemplo que el conjunto OSULeaf representa un esquema unidimensional de hojas de árboles con formas de puntas en sus terminaciones, ver figura 3.5, al extraer los bordes, las series de tiempo presentan cambios abruptos. El conjunto Synthetic-Control también presenta series de tiempo con cambios fuertes, con puntos angulosos, figura 3.6, lo que permite entrever que los métodos propuestos presentan dificultad al tratar series de tiempo que presentan cambios abruptos en su dinámica. El kernel base utilizado para realizar los embebimientos en el RKHS fue el kernel RBF el cual puede presentar mejores rendimientos para series de tiempo con cambios suaves y fácilmente diferenciables. El artículo [8] apoya lo anterior, respecto a trabajar con kernel RBF y obtener una alta tasa de error con la serie OSULeaf. Para reforzar estos resultados se usó el concepto de Coeficiente de Variación (CV) multivariado, que mide la variabilidad de un conjunto de series temporales. De acuerdo con [61] si se tiene un conjunto de L series de tiempo de longitud N, con vector medio $\mu \neq 0$ y matriz de covarianza Σ, el CV multivariado se define como

$$CV = (Tr(\Sigma)(\mu^T\mu)^{-1})^{1/2} \; donde \qquad (4.1)$$
$$Tr(\Sigma) \; es \; la \; traza \; de \; la \; matriz \; \Sigma$$

Tabla 4.2: Conjunto de datos donde CV es mayor que 2

Nombre conjunto	**CV conjunto entrenamiento**	**CV conjunto de prueba**	**SVM KMMD**	**SVM KDC_(t)**
50words	3,0958	3,1389	0,701	0,310 (2)
OSULeaf	2,8892	2,8740	0,525	0,426 (1)
synthetic_control	12,0300	11,6455	0,510	0,360 (1)

En la tabla 4.2 se puede observar que cuando el CV del conjunto de entrenamiento y el conjunto de prueba de un conjunto de datos dado es alto (mayor que 2), la tasa de error de los métodos propuestos es alta, por ejemplo el conjunto Synthetic-Control. Cabe señalar que lo anterior debe ser confirmado realizando mayores experimentos con conjuntos de series de tiempo que presenten las características señaladas.

La figura 4.2d compara a SVM-KDC con BoP. Los resultados observados permiten decir que no hay una buena capacidad de generalización en los métodos dado que se desempeñan bien para unas series pero para otras no. Al revisar la fila Gana en la tabla 4.1 no hay ningún método que gane al menos sobre el 50 % de los conjuntos de datos estudiados. Además de acuerdo a las pruebas estadísticas las diferencias en los rendimientos promedios, fila Media tabla 4.1, no son estadísticamente significantes. Al revisar el promedio de todos los conjuntos de datos, el clasificador SVM-KDC pierde contra el clasificador 1-NN DTW debido al conjunto Synthetic-Control, dado que, como se mencionó anteriormente al tener un CV alto, da una tasa de error alta respecto a los otros clasificadores, pero si se hace el análisis sin tener en cuenta la tasa de este conjunto, el clasificador SVM-KDC presenta la menor tasa de error promedio sobre todos los conjuntos de datos junto a 1-NN DTW.

Otro análisis importante es revisar el resultado de los vectores de soporte (SV), que en este caso serían distribuciones de soporte en el RKHS, para cada conjunto de datos. La tabla 4.3 contiene la cantidad de SV por clasificador y conjunto de datos. En ella se aprecia una alta cantidad de SV respecto al conjunto de entrenamiento para los conjuntos multiclase, lo cual puede deberse a la holgura permitida en la búsqueda de los hiperparámetros y a la distribución de los datos en el espacio RKHS. El alto número de SV indica que las clases no son fácilmente separables o los datos de entrenamiento por clase son insuficientes. Para conjuntos con solo dos clases como ECG200, Coffee y Gun_Point el número de SV respecto al conjunto de entrenamiento es más bajo, indicando una mayor separabilidad entre las clases.

Tabla 4.3: Cantidad de SV por clasificador y conjunto de datos

Nombre conjunto	Número de clases	Conjunto de entrenamiento	Cantidad SV SVM-KDC	Cantidad SV SVM-KMMD
50words	50	450	420	432
Adiac	37	390	339	295
Beef	5	30	23	28
CBF	3	30	22	27
Coffee	2	28	11	16
ECG200	2	100	33	32
FISH	7	175	117	118
FaceFour	4	24	22	16
Gun_Point	2	50	16	17
Lighting2	2	60	43	22
Lighting7	7	70	62	60
OSULeaf	6	200	188	186
OliveOil	4	30	18	23
SwedishLeaf	15	500	355	400
Trace	4	100	60	27
synthetic_control	6	300	269	264

Capítulo 5

Conclusiones y trabajos futuros

En este trabajo se adoptan dos métodos de clasificación basados en el embebimiento de distribuciones de probabilidad en un RKHS. Se implementa el clasificador SVM-KMMD el cual asume que todos los aspectos en los datos son importantes modelando las distribuciones marginales, sus resultados no presentaron una buena generalización sobre los 16 conjuntos de datos de series de tiempos UCR estudiadas, presentando mayores tasas de error que los otros clasificadores presentados. El otro método llamado SVM-KDC al contrario presentó un rendimiento interesante comparado con métodos del estado del arte como 1NN-ED y 1NN-DTW y mejoró el rendimiento medio de los clasificadores basados en características como FBL y TFRP. El método SVM-KDC asume que sólo ciertos aspectos en los datos tienen influencia en la respuesta del clasificador, por lo tanto modela las distribuciones condicionales permitiendo explotar la dependencia entre los datos y obteniendo bajas tasas de error.

El métodos SVM-KDC se presenta como una buena alternativa para la clasificación de series de tiempo en contra propuesta a los métodos de representación basados en características, los cuales como se observó pueden presentar problemas al momento de seleccionar las características discriminantes entre las diferentes clases de series de tiempo en un dominio de aplicación, y más cuando las clases no se encuentran balanceadas y presentan una gran heterogeneidad en el tamaño del conjunto de entrenamiento.

Se comprueba la importancia de modelar las dependencias entre los datos en problemas jerárquicos o espacio temporales como es el caso de las series de tiempo. El clasificador SVM-KDC presenta mejores tasas de error que el modelo SVM-KMMD al modelar las dependencia en los datos a partir de las distribuciones condicionales.

El clasificador SVM-DC presenta buen rendimiento al trabajar sobre problemas multiclase. Diferentes desarrollos sobre métricas para la clasificación de series de tiempo, tienden a probar sus modelos sobre solo problemas de dos clases, mientras el clasificador propuesto presentó bajas tasas de error para conjuntos multiclase, por ejemplo SwedishLeaf con 15 clases y Beef con 5 clases.

Para futuros trabajos sería interesante buscar una forma más sustentada para encontrar los hiperparámetros de los clasificadores SVM-KMMD y SVM-KDC, debido a que en este trabajo se encontraron de forma experimental armando las cuadrículas dentro de los rangos establecidos. También podría revisarse si existe una mejora en las tasas de error al buscar los parámetros con pasos más finos alrededor de los parámetros ya encontrados en estos experimentos.

Desarrollos futuros también podrían revisar el rendimiento de los clasificadores con kernel diferentes al kernel RBF, el cual mostró para series de tiempo con cambios abruptos altas tasas de error. También se abre la posibilidad de probar combinaciones de kernels de tal manera que se logre obtener un clasificador con mayor generalización. El clasificador SVM-KDC podría hacer parte de un algoritmo de ensamble.

En relación al costo computacional, se señala que ambos métodos son fácilmente paralelizables (la matriz kernel puede ser dividida en varias submatrices, y las sumas pertinentes se calculan de forma independiente antes de combinarse). Para mitigar los altos costos computacionales, en [51] proponen el framework de características aleatorias de fourier (RFF), y en [62] mencionan que podrían utilizarse métodos aleatorios para acelerar los dobles ciclos requeridos en los algoritmos, por ejemplo para el cálculo de MMD, computando solo partes de la suma, sin embargo estos procedimientos reducirían la calidad del estimador.

Bibliografía

[1] Bing Hu, Yanping Chen, and Eamonn Keogh. Time series classification under more realistic assumptions. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 578–586. SIAM, 2013.

[2] Basabi Chakraborty. Feature selection and classification techniques for multivariate time series. In *Innovative Computing, Information and Control, 2007. ICICIC'07. Second International Conference on*, pages 42–42. IEEE, 2007.

[3] Young-Seon Jeong and Raja Jayaraman. Support vector-based algorithms with weighted dynamic time warping kernel function for time series classification. *Knowledge-based systems*, 75:184–191, 2015.

[4] Tak-chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[5] Joan Serra and Josep Ll Arcos. An empirical evaluation of similarity measures for time series classification. *Knowledge-Based Systems*, 67:305–314, 2014.

[6] Eamonn Keogh. A decade of progress in indexing and mining large time series databases. In *Proceedings of the 32nd international conference on Very large data bases*, pages 1268–1268. VLDB Endowment, 2006.

[7] Liqiang Pan, Qi Meng, Wei Pan, Yi Zhao, and Huijun Gao. A feature segment based time series classification algorithm. In *Instrumentation and Measurement, Computer, Communication and Control (IMCCC), 2015 Fifth International Conference on*, pages 1333–1338. IEEE, 2015.

[8] Lei Liu, Wangmeng Zuo, David Zhang, and Dongyu Zhang. Learning with multiple gaussian distance kernels for time series classification. In *Advanced Computer Control (ICACC), 2011 3rd International Conference on*, pages 624–628. IEEE, 2011.

[9] Penugonda Ravikumar and V Susheela Devi. Weighted feature-based classification of time series data. In *Computational Intelligence and Data Mining (CIDM), 2014 IEEE Symposium on*, pages 222–228. IEEE, 2014.

[10] Vinicius MA Souza, Diego F Silva, and Gustavo EAPA Batista. Extracting texture features for time series classification. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 1425–1430. IEEE, 2014.

[11] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, pages 1–35, 2013.

[12] Christos Faloutsos, Mudumbai Ranganathan, and Yannis Manolopoulos. *Fast subsequence matching in time-series databases*, volume 23. ACM, 1994.

[13] Flip Korn, Hosagrahar V Jagadish, and Christos Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *Acm Sigmod Record*, volume 26, pages 289–300. ACM, 1997.

[14] Kin-Pong Chan and Ada Wai-Chee Fu. Efficient time series matching by wavelets. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 126–133. IEEE, 1999.

[15] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.

[16] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Sigmod Record*, 30(2):151–162, 2001.

[17] Yuhan Cai and Raymond Ng. Indexing spatio-temporal trajectories with chebyshev polynomials. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 599–610. ACM, 2004.

[18] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.

[19] QiuxiaChen LeiChen XiangLian YunhaoLiu and Jeffrey Xu Yu. Indexable pla for efficient similarity search. 2007.

[20] Jessica Lin, Rohan Khade, and Yuan Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 39(2):287–315, 2012.

[21] Alex Nanopoulos, Rob Alcock, and Yannis Manolopoulos. Feature-based classification of time-series data. *International Journal of Computer Research*, 10(3):49–61, 2001.

[22] Xiaozhe Wang, Kate Smith, and Rob Hyndman. Characteristic-based clustering for time series data. *Data mining and knowledge Discovery*, 13(3):335–364, 2006.

[23] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153, 2013.

[24] Xiaozhe Wang, Anthony Wirth, and Liang Wang. Structure-based statistical features and multivariate time series clustering. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 351–360. IEEE, 2007.

[25] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[26] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.

[27] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. Discovering similar multidimensional trajectories. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 673–684. IEEE, 2002.

[28] Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 792–803. VLDB Endowment, 2004.

[29] Lei Chen, M Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502. ACM, 2005.

[30] Elias Frentzos, Kostas Gratsias, and Yannis Theodoridis. Index-based most similar trajectory search. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 816–825. IEEE, 2007.

[31] Michael D Morse and Jignesh M Patel. An efficient and accurate method for evaluating time series similarity. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 569–580. ACM, 2007.

[32] Yueguo Chen, Mario A Nascimento, Beng Chin Ooi, and Anthony KH Tung. Spade: On shape-based pattern detection in streaming time series. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 786–795. IEEE, 2007.

[33] Fabian Mörchen. Time series feature extraction for data mining using dwt and dft, 2003.

[34] Yi-Leh Wu, Divyakant Agrawal, and Amr El Abbadi. A comparison of dft and dwt based similarity search in time-series databases. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 488–495. ACM, 2000.

[35] Karen Zita Haigh, Wendy Foslien, and Valerie Guralnik. Visual query language: Finding patterns in and relationships among time series data. In *Proceedings of the seventh Workshop on Mining Scientific and Engineering Datasets*, 2004.

[36] Ruoqian Liu and Yi L Murphey. Time-series temporal classification using feature ensemble learning. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–5. IEEE, 2010.

[37] Ben D Fulcher and Nick S Jones. Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3026–3037, 2014.

[38] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[39] Maya Kallas, Paul Honeine, Clovis Francis, and Hassan Amoud. Kernel autoregressive models using yule–walker equations. *Signal Processing*, 93(11):3053–3061, 2013.

[40] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*, pages 13–31. Springer, 2007.

[41] Le Song, Jonathan Huang, Alex Smola, and Kenji Fukumizu. Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 961–968. ACM, 2009.

[42] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*, pages 13–31. Springer, 2007.

[43] Arash Ali Amini. *High-dimensional principal component analysis*. University of California, Berkeley, 2011.

[44] Kenji Fukumizu, Francis R Bach, and Michael I Jordan. Kernel dimensionality reduction for supervised learning. In *Advances in Neural Information Processing Systems*, pages 81–88, 2004.

[45] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Scholkopf. Measuring statistical dependence with hilbert-schmidt norms. In *ALT*, volume 16, pages 63–78. Springer, 2005.

[46] Arthur Gretton, Kenji Fukumizu, Zaid Harchaoui, and Bharath K Sriperumbudur. A fast, consistent kernel two-sample test. In *Advances in neural information processing systems*, pages 673–681, 2009.

[47] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[48] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[49] Le Song, Kenji Fukumizu, and Arthur Gretton. Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *IEEE Signal Processing Magazine*, 30(4):98–111, 2013.

[50] Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520, 2007.

[51] Jovana Mitrovic, Dino Sejdinovic, and Yee-Whye Teh. Dr-abc: approximate bayesian computation with kernel-based distribution regression. In *International Conference on Machine Learning*, pages 1482–1491, 2016.

[52] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. `www.cs.ucr.edu/~eamonn/time_series_data/`.

[53] Toni M Rath and Raghavan Manmatha. Word image matching using dynamic time warping. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2003.

[54] E. Bagnall, A. Keogh. Repositorio UEA, UCR. `http://timeseriesclassification.com/description.php?Dataset=FiftyWords`, 2017. [En línea; disponible 19-noviembre-2017].

[55] E. Bagnall, A. Keogh. Repositorio UEA, UCR. `http://www.timeseriesclassification.com/description.php?Dataset=Adiac`, 2017. [En línea; disponible 19-noviembre-2017].

[56] E. Bagnall, A. Keogh. Repositorio UEA, UCR. `http://www.timeseriesclassification.com/description.php?Dataset=FaceFour`, 2017. [En línea; disponible 19-noviembre-2017].

[57] E. Bagnall, A. Keogh. Repositorio UEA, UCR. `http://www.timeseriesclassification.com/description.php?Dataset=OSULeaf`, 2017. [En línea; disponible 19-noviembre-2017].

[58] Robert J Alcock, Yannis Manolopoulos, et al. Time-series similarity queries employing a feature-based approach. In *7th Hellenic conference on informatics*, pages 27–29, 1999.

[59] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[60] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[61] LEIGH Van Valen. The statistics of variation. *Variation: A central concept in biology*, pages 29–48, 2005.

[62] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.